LPSA: Leaf-Prior Structural Attack on Graph Neural Networks at Scale

Yu Ran*, Chenbo Ma*, Shiqi Liu*, Zihan Chen, Yi Pan, Xin Zhang, Maoyi Xiong, Wentao Zhao†
National University of Defense Technology
{ranyu, machenbo, liushiqi, chenzihan21, panyi_jsjy, zhang_xin, xiongmaoyi, wtzhao}@nudt.edu.cn

***Abstract*—Adversarial attacks on Graph Neural Networks (GNNs) have exposed critical robustness vulnerabilities in graph-based learning. However, existing attack methods face a fundamental trade-off between effectiveness and scalability: exhaustive greedy attacks achieve strong effectiveness on small graphs but become computationally intractable at scale, while scalable optimization-based attacks suffer from slow convergence, requiring excessive computational overhead to achieve effective perturbations. To bridge this gap, we propose Leaf-Prior Structural Attack (LPSA), a simple yet efficient and effective structural attack for large-scale graphs. LPSA is built upon two key insights. First, structural asymmetry in graphs makes leaf nodes highly effective adversarial injection points, enabling strong influence on the target node's prediction with minimal structural modifications. Leveraging this, LPSA restricts the perturbation search space to a compact leaf-prior subspace, dramatically reducing the computational overhead of structural attacks. Second, inductive biases and gradient misalignment in surrogate models degrade attack transferability. To address this issue, we introduce a manifold-smoothed surrogate fine-tuning strategy that regularizes the surrogate's gradient field to align with the intrinsic data distribution, thereby obtaining more transferable adversarial gradients. By combining leaf-prior search with single-step gradient-based perturbation generation, LPSA eliminates costly iterative optimization while maintaining strong attack effectiveness. Extensive experiments on four benchmark datasets, including the million-scale MAG dataset, demonstrate that LPSA not only achieves competitive or superior attack success rates but also reduces runtime by over 80% compared to state-of-the-art baselines, particularly on large-scale graphs. The source code is available at https://github.com/rannyu/LPSA.**

***Index Terms*—Large-scale Graph, Structural Attack, Graph Neural Networks**

I. Introduction

Graph Neural Networks (GNNs) have demonstrated strong performance on node classification by jointly leveraging node attributes and graph topology, and have consequently been applied across a broad range of domains, including social networks [1], bioinformatics [2], and communication systems [3]. However, their reliance on message passing also introduces a critical vulnerability. Specifically, even minor structural perturbations can propagate through neighborhoods and be amplified, leading to severe misclassification and raising serious concerns regarding robustness and security [4].

Adversarial attacks on graphs remain effective even when the attacker possesses limited knowledge [5], [6], [4], [7]. In particular, gray-box local evasion attacks pose a more significant threat in deployed systems, where the attackers have no access to model parameters nor interfere with training [8]. Specifically, an attacker typically targets a specific node at test time and misleads target models by modifying a small number of edges, while preserving the utility of the graph and avoiding creating conspicuous global anomalies. Existing methods heavily rely on iterative edge perturbations on surrogate models, which require repeated gradient updates, re-scoring of candidate edges, or combinatorial search [9]. These repetitive operations incur substantial computational overhead and exhibit limited transferability across architectures.

To address these limitations, we identify a structural prior intrinsic to message-passing GNNs—leaf nodes. Owing to their asymmetric role in neighborhood aggregation, leaf nodes serve as structurally isolated yet influential injection points, enabling targeted perturbations with predictable impact on the target node's representation. This insight suggests that effective local attacks can focus on a compact, structurally privileged subset rather than searching the full candidate space.

In this paper, we propose Leaf-Prior Structural Attack (LPSA), a highly efficient framework designed for adversarial attacks on large-scale graphs. LPSA adopts a two-stage strategy: (i) training a manifold-smoothed surrogate model to improve gradient transferability; and (ii) performing a single-step leaf-prior structural perturbation to induce a substantial logit shift on the target node. Specifically, in stage (i), we augment standard supervised training with mechanisms that encourage smooth and consistent representations, thereby mitigating surrogate overfitting to a particular architecture and improving cross-model gradient alignment. In stage (ii), we replace iterative optimization with a single-step attack by restricting the search space to a leaf-induced candidate block. Subject to a degree-adaptive perturbation budget, we select adversarial edges incident to the target node in a single shot by maximizing an adversarial margin objective. This approach achieves a lightweight yet effective attack. Comprehensive experiments on various benchmarks validate the effectiveness and scalability of LPSA. The results show that LPSA reduces computational overhead by approximately 80%.

The contribution of this paper is three-fold:

- We propose LPSA, which combines manifold-smoothed surrogate training with a leaf-prior, single-step edge perturbation strategy, thereby improving both attack effi-

* indicates equal contributions.
† indicates corresponding author.

ciency and cross-architecture transferability without relying on iterative optimization.
- We reveal and characterize the leaf-node structural asymmetry in message-passing GNNs, showing that leaf nodes constitute an efficient injection point for local structural perturbations and provide an interpretable prior for pruning the attack search space.
- We conduct systematic evaluations under the gray-box local evasion setting, including comparisons against representative state-of-the-art baselines and necessary ablations, and demonstrate that LPSA achieves competitive attack success rates with reduced computational cost.

II. Related Work

Adversarial Attacks on GNNs. Graph Neural Networks (GNNs) have achieved strong performance in a wide range of graph learning tasks, but have also been shown to be vulnerable to adversarial perturbations [4], [10], [7]. Graph adversarial attacks aim to mislead a trained model by introducing carefully crafted and imperceptible perturbations to the input graph. Existing attack methods mainly manipulate graph structure or node attributes, including adding or removing edges [11], modifying node features [12], and injecting fake nodes [13]. Representative approaches range from greedy searches like Nettack [4] and heuristics like DICE [11] to gradient-based relaxations that iteratively optimize continuous edge variables [12]. While these methods demonstrate strong effectiveness on small graphs, they typically rely on expensive gradient computation or combinatorial search, which makes them difficult to scale to large graphs.

Adversarial Attacks on Large-Scale Graphs. Real-world applications often involve million-scale graphs where significant computational and memory costs make standard gradient computation and iterative optimization difficult. To address these challenges, recent work has proposed scalable attack strategies by restricting the optimization space to tractable substructures. PR-BCD [9] reduces memory consumption by iteratively optimizing randomly sampled blocks of candidate edges, while its variant GR-BCD further improves efficiency through greedy selection within each block. Similarly, SGA [14] limits gradient computation to a local k-hop neighborhood around the target node, significantly reducing time and memory costs. Despite these improvements, fundamental limitations remain. Existing scalable attacks still rely on iterative optimization to find effective perturbations, requiring multiple gradient updates and incurring significant runtime overhead. This iterative nature fundamentally limits their scalability and practicality for attacking real-world, million-scale graphs in time-sensitive scenarios. Moreover, they treat candidate edges uniformly, ignoring the intrinsic structural asymmetry of graphs, particularly the unique vulnerability of leaf nodes in message-passing GNNs, which we theoretically analyze and exploit. In contrast, we propose LPSA, which introduces a leaf-prior structural bias and a manifold-smoothed surrogate to enable highly efficient single-step attacks. This approach bypasses costly iterative optimization while maintaining strong transferability, offering a new paradigm for scalable graph adversarial attacks.

TABLE I: Summary of notations.

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$	Undirected attributed graph
N, D, C	Number of nodes, feature dimensions and classes
$\mathbf{A} \in \{0,1\}^{N \times N}$	Adjacency matrix
$\mathbf{X} \in \mathbb{R}^{N \times D}$	Node feature matrix
$\mathcal{Y} = \{1, \ldots, C\}$	Label set
$\deg(v_i)$	Degree of node v_i
$\mathcal{F}(\cdot)$	Victim model
$f(\cdot)$	Surrogate model
v_t	Target node
$\mathbf{Z} \in \mathbb{R}^{N \times C}$	Logits of the surrogate model $f(\cdot)$
Δ	Perturbation budget (max number of modified edges)
α	Perturbation ratio ($\Delta = \text{round}(\deg(v_t) \cdot \alpha)$)
$\mathcal{S}_{leaf}$	Leaf-prior search space

III. Preliminaries

To facilitate the discussion, we formally define the problem of node classification and specify the threat model considered in this work. Key notations are listed in Table I.

Notations. We denote an undirected attributed graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The topological structure is represented by the adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$, where $\mathbf{A}_{ij} = 1$ indicates a connection between v_i and v_j. The node attributes are denoted by $\mathbf{X} \in \mathbb{R}^{N \times D}$. In this paper, we represent the graph interchangeably as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ or $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ depending on the context.

Node Classification. GNNs learn node representations by recursively aggregating information from local neighborhoods. For the node classification task, given a graph $\mathcal{G}$ with labeled nodes $\mathcal{V}_L \subseteq \mathcal{V}$ and unlabeled nodes $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_L$, each node $v_i \in \mathcal{V}_L$ is associated with a ground-truth label $y_i \in \mathcal{Y}$, where $\mathcal{Y} = \{1, \ldots, C\}$ denotes the set of C classes. Node classification aims to learn a predictive mapping $\mathcal{F}(\mathbf{A}, \mathbf{X}) \to \mathbb{R}^{N \times C}$ to minimize the prediction error on $\mathcal{V}_L$. Formally, let $\mathbf{Z} = f_\theta(\mathbf{A}, \mathbf{X}) \in \mathbb{R}^{N \times C}$ denote the output logits of a (surrogate) GNN model f_θ, where $\mathbf{Z}_{i,c}$ is the logit for node v_i and class c. The predicted probability distribution is obtained via the softmax function: $\mathbf{P}_i = \text{softmax}(\mathbf{Z}_i) \in \mathbb{R}^C$. The predicted label is then $\hat{y}_i = \arg\max_c \mathbf{P}_{i,c}$.

Threat Model. Existing attacks can be categorized based on the adversary's knowledge (White-box vs. Black/Gray-box), the attack scope (Global vs. Local), and the attack stage (Poisoning vs. Evasion). In this work, we specifically focus on the Gray-box Local Evasion Attack, which is the most practical yet challenging scenario in real-world applications. This setting entails: (i) Knowledge: The adversary has access to the training data and the victim model's final decisions, but not to the internal parameters or gradients. (ii) Scope: Induce misclassification on specific target nodes (Local) rather than degrading the overall system performance. (iii) Timing: Perturbations are applied during the test phase (Evasion).

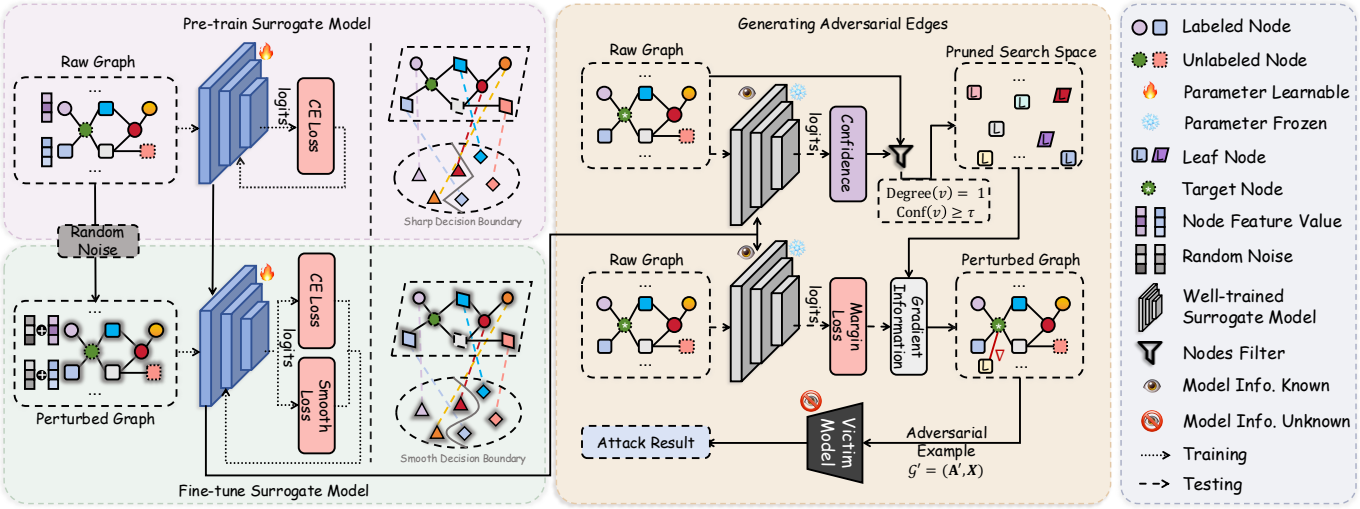


Fig. 1: Overview of **LPSA**. The framework proceeds in two stages: **(Left) Manifold-Smoothed Surrogate Training**, where we add Gaussian noise and an explicit smooth loss to align the decision boundary with the data manifold; and **(Right) Leaf-Prior Structural Attack**, where we exploit a leaf-prior strategy to prune the search space and generate adversarial edges via a single gradient calculation, transferring the result to the victim model.

IV. Proposed Method

A. Problem Definition

Attacker's Capability. We assume that the adversary has access to the training data (i.e., graph structure $\mathcal{G}$ and node features $\mathbf{X}$), while having no knowledge of the victim model's specific architecture and trained parameters. Under this restrictive setting, the adversary typically trains a surrogate model f using the available data. The adversarial perturbations are then generated based on f without querying the victim model, relying on the *transferability* of adversarial examples to mislead the target model $\mathcal{F}$.

Furthermore, to ensure the *imperceptibility* of the attack, the structural perturbations must be subtle and unnoticeable. To satisfy this property, the modification of the edge set $\mathcal{E}$ is strictly limited by a perturbation budget Δ. Specifically, the number of modified edges is constrained relative to the degree of the target node v_t, defined as $\Delta = \text{round}(\deg(v_t) \cdot \alpha)$, where α is a small constant controlling the maximum perturbation ratio.

Attacker's Goal. Given a well-trained victim GNN model $\mathcal{F}$ and a target node v_t with its ground-truth label y_t, the attacker aims to craft a perturbed adjacency matrix $\mathbf{A}'$ such that the model predicts any class other than y_t. Formally, the objective is to construct a perturbation $\mathbf{A}'$ such that:

$$\arg\max_{c \in \mathcal{Y}} \left[\text{softmax}\left(\mathcal{F}(\mathbf{A}', \mathbf{X})_{v_t}\right)\right]_c \neq y_t, \quad \text{s.t. } \|\mathbf{A}' - \mathbf{A}\|_0 \leq \Delta, \tag{1}$$

where $\mathcal{F}(\mathbf{A}', \mathbf{X})_{v_t} \in \mathbb{R}^C$ denotes the output logits of the victim model for the target node v_t. The constraint $\|\mathbf{A}' - \mathbf{A}\|_0 \leq \Delta$ ensures that the graph structure is only slightly modified by the adversary, preserving the imperceptibility of the attack. In this work, following the local attack paradigm, the perturbations are restricted to edges directly connected to the target node v_t.

B. Overview

We propose LPSA (**L**eaf-**P**rior **S**tructural **A**ttack), a methodology which exploits topological leaf priors to facilitate efficient and transferable local attacks. An illustrative overview of the framework is provided in Fig. 1. In our attack, a two-phase strategy is adopted for gray-box local evasion attack. During the surrogate model training phase, we align the decision boundary with the data manifold to enhance transferability. In the subsequent attack phase, we implement a leaf-prior structural attack on this robust surrogate to generate perturbations, which are then transferred to the victim model. In the following, we elaborate on each key component of our framework.

C. Manifold-Smoothed Surrogate Model Training

Pre-training. Following the established transfer-based attack paradigm, we first pre-train the surrogate model f_θ on the labeled node set $\mathcal{V}_L$ to capture discriminative features:

$$\mathcal{L}_{CE} = -\sum_{v_i \in \mathcal{V}_L} \mathbf{y}_i \log(f_\theta(\mathbf{A}, \mathbf{X})_i), \tag{2}$$

where the learned parameters serve as the foundation for the subsequent fine-tuning phase.

Fine-tuning. Inspired by the principles of Score Matching [15] and distribution-based transferability [16], a transferable adversarial gradient should align with the intrinsic data distribution. To foster this alignment, we first inject Gaussian noise $\delta \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ into the input features to generate perturbed samples $\tilde{\mathbf{X}} = \mathbf{X} + \delta$. We then introduce a smoothing loss ($\mathcal{L}_{smooth}$) to penalize the squared norm of the loss gradient with respect to these perturbed inputs. To further refine the gradient signal, a temperature parameter T

is incorporated to soften the probability distribution, enabling the gradient to capture more informative inter-class signals:

$$\mathcal{L}_{smooth} = \mathbb{E}_{\mathbf{X},\delta}\left[\left\|\nabla_{\tilde{\mathbf{X}}} \log \text{softmax}\left(\mathbf{Z}(\tilde{\mathbf{X}})/T\right)_y\right\|_2^2\right], \quad (3)$$

where $\mathbf{Z}(\tilde{\mathbf{X}}) = f_\theta(\mathbf{A}, \tilde{\mathbf{X}})$ denotes the logits under perturbed features. The final joint optimization objective is formulated as $\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{L}_{smooth}$, where λ governs the trade-off between classification accuracy and geometric smoothness. We optimize this objective to obtain the final surrogate model parameters θ^*, which are used in the subsequent attack phase.

Theoretical Analysis. The effectiveness of $\mathcal{L}_{smooth}$ can be interpreted using a first-order Taylor expansion. For a minor perturbation δ, the loss at $\mathbf{X} + \delta$ can be approximated as:

$$\mathcal{L}(\mathbf{X} + \delta) \approx \mathcal{L}(\mathbf{X}) + \nabla_{\mathbf{X}}\mathcal{L}(\mathbf{X})^T\delta. \quad (4)$$

To ensure the generated gradient aligns with the intrinsic data manifold, the surrogate should exhibit local smoothness, implying that the variation $\mathcal{L}(\mathbf{X} + \delta) - \mathcal{L}(\mathbf{X})$ should be minimized. By penalizing the squared gradient norm $\|\nabla_{\mathbf{X}}\mathcal{L}\|_2^2$, we directly compress the first-order sensitivity of the surrogate. Geometrically, this encourages the decision boundary to conform to the underlying data manifold, promoting gradients that capture the intrinsic data distribution rather than overfitting to model-specific artifacts.

D. Leaf-Prior Structural Attack

Leveraging the manifold-smoothed surrogate, we propose a streamlined attack strategy that exploits the *topological vulnerability* of graph structures. We consider that blind searches over the global edge space are inefficient. Instead, we identify **leaf nodes** (degree $d = 1$) as the optimal perturbation candidates for maximizing the marginal adversarial impact.

Structural Vulnerability: The Leaf Node Advantage. Leaf nodes (degree-1 nodes) exhibit a unique structural vulnerability in message-passing GNNs. Unlike high-degree nodes, their embeddings are largely determined by their own features with minimal neighbor interference. This makes them highly effective carriers for adversarial perturbations.

Formally, consider a target node v_t with mean aggregation. For simplicity, we consider neighbor-only aggregation; the argument extends naturally to common variants with self-loops. When an adversary connects a leaf node v_ℓ to v_t, the aggregated representation shifts to:

$$h_t' = \frac{|N_t|}{|N_t|+1}\bar{h}_t + \frac{1}{|N_t|+1}h_\ell, \quad (5)$$

where $\bar{h}_t$ denotes the mean embedding of v_t's original neighbors and $|N_t|$ represents the degree of v_t. This formulation reveals a critical structural asymmetry:

- **Low Neighborhood Interference.** Since v_ℓ is a leaf node, its representation $h_\ell = \phi(\mathbf{W} \cdot \mathbf{x}_\ell)$ is primarily determined by its own features, with minimal influence from neighborhood aggregation (where ϕ is the activation function). Unlike interior nodes, this property prevents the injection signal from being diluted by message-passing noise. As a result, if v_ℓ is confidently predicted as a counter-class, it serves as a precise and effective adversarial injector.
- **Directional Injection:** By strategically selecting leaf nodes from a counter-class with high prediction confidence, connecting it induces a deterministic shift in the target node's latent space. This injection of a potent conflicting signal disrupts the target's original latent cluster and significantly diminishes the logit margin of the ground truth, effectively pushing the target node to cross the decision boundary toward the non-target region.

Leaf-Prior Search Space Pruning. Based on the insight of structural asymmetry, we restrict the search space from the global candidate set to a subset of leaf nodes (termed a *block*). Specifically, we select leaf nodes predicted by the surrogate model as counter-class with high confidence. Formally, let $\mathcal{S}_{pool}$ denote the candidate pool of threatening nodes:

$$\mathcal{S}_{pool} = \{v_j \in \mathcal{V} \setminus \{v_t\} \mid \deg(v_j) = 1,\ \hat{y}_j \neq y_t,\ p_{\theta^*}(\hat{y}_j|v_j) > \tau\}, \quad (6)$$

where $\hat{y}_j$ denotes the predicted class of node v_j, and $p_{\theta^*}(\hat{y}_j|v_j)$ represents the corresponding prediction confidence.

To ensure the scalability of the attack on large-scale graphs, we impose a block size constraint B. The final search space $\mathcal{S}_{leaf}$ is constructed by selecting the top-B nodes from $\mathcal{S}_{pool}$ based on their prediction confidence:

$$\mathcal{S}_{leaf} = \text{Top-B}(\mathcal{S}_{pool}), \quad \text{sorted by } p_{\theta^*}(\hat{y}_j|v_j). \quad (7)$$

By focusing on the optimization to these structurally isolated yet aggressive nodes, we effectively compress the search space to the most critical decision boundaries.

Scope and Limitations. We note that the effectiveness of the leaf-prior strategy relies on the presence of a non-trivial number of degree-one nodes. Empirically, many real-world graphs (e.g., citation and web graphs) exhibit heavy-tailed degree distributions with abundant leaf nodes. For graphs with extremely dense or regular structures, the candidate pool may shrink, in which case LPSA reduces to a constrained local search without leaf prioritization, which preserves correctness while trading off part of its efficiency advantage.

Attack Procedure. The technical details are summarized in Algorithm 1. We first narrow the perturbation search space from the global set of candidate edges to a restricted set induced by leaf nodes $\mathcal{S}_{leaf}$. To explicitly mislead the target node, we formulate the edge generation as an optimization problem that maximizes the adversarial margin loss $\mathcal{L}_{adv}$:

$$\mathcal{L}_{adv} = \max_{j \neq y_t} \mathbf{Z}_{t,j} - \mathbf{Z}_{t,y_t}, \quad (8)$$

where $\max_{j \neq y_t} \mathbf{Z}_{t,j}$ corresponds to the logit of the most competitive incorrect class, i.e., the non-ground-truth class with the highest prediction score. Maximizing $\mathcal{L}_{adv}$ increases the margin between the most competitive non-target class and the ground truth y_t, effectively pushing the target node across the decision boundary. Since the discrete graph structure is non-differentiable, we adopt a continuous relaxation with a

Algorithm 1 LPSA: Leaf-Prior Structural Attack

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, target node v_t, perturbation budget Δ, surrogate model f_θ.
Parameters: Noise scale σ, temperature parameter T, confidence threshold τ, block size B, trade-off parameter λ.
Output: Adversarial adjacency matrix $\mathbf{A}'$

procedure TRAINING SURROGATE MODEL
  Pre-train f_θ minimizing $\mathcal{L}_{CE}$ on labeled nodes $\mathcal{V}_L$;
  Sample Gaussian noise $\delta \sim \mathcal{N}(0, \sigma^2\mathbf{I})$;
  $\tilde{\mathbf{X}} \leftarrow \mathbf{X} + \delta$;
  Fine-tune f_θ minimizing $\mathcal{L}_{total}$ on $\tilde{\mathbf{X}}$ with T;
  return refined surrogate model f_{θ^*};

procedure PERFORM ATTACK ON v_t
  Construct $\mathcal{S}_{pool}$ with Eq. (6);
  $\mathcal{S}_{leaf} \leftarrow$ Top-$B(\mathcal{S}_{pool})$ based on confidence (Eq. (7));
  Construct relaxed adjacency $\tilde{\mathbf{A}} \leftarrow \mathbf{A} + \mathbf{P}$;
  Compute logits $\mathbf{Z}_t \leftarrow f_{\theta^*}(\tilde{\mathbf{A}}, \mathbf{X})_{v_t}$;
  $\mathcal{L}_{adv} \leftarrow \max_{j \neq y_t} \mathbf{Z}_{t,j} - \mathbf{Z}_{t,y_t}$;
  Compute gradient $\mathbf{g} \leftarrow \nabla_{\mathbf{P}} \mathcal{L}_{adv}$;
  Select top-Δ edges in $\mathcal{S}_{leaf}$ according to $|\mathbf{g}|$;
  Update $\mathbf{A}'$ with the selected edges;
  return Adversarial adjacency matrix $\mathbf{A}'$;

perturbation matrix $\mathbf{P}$. Importantly, $\mathbf{P}$ is defined only over the candidate edges in $\mathcal{S}_{leaf}$, rather than the full adjacency. To enable gradient computation, we define a relaxed adjacency matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{P}$. We compute the gradient $\nabla_{\mathbf{P}} \mathcal{L}_{adv}$ via backward propagation on the surrogate model, and restrict gradient computation to the continuous perturbation variables associated with candidate leaf nodes in $\mathcal{S}_{leaf}$. The final discrete perturbation is obtained by selecting the top-Δ entries in $\mathbf{P}$ with the largest gradient magnitudes within the restricted space. These selected perturbations are then applied to generate the adversarial graph, which is subsequently transferred to the victim models. Notably, the entire attack requires only a single backward propagation, leading to substantially reduced computational overhead while maintaining strong transferability across different GNN architectures.

E. Complexity Analysis

We analyze the computational complexity of LPSA, emphasizing its scalability and memory efficiency. In the training phase, the surrogate model is pre-trained and fine-tuned for a small number of epochs, incurring minimal overhead compared to standard GNN training. We focus here on the attack-phase complexity. **Space Complexity:** (i) *Graph Storage:* Using sparse representations, the adjacency matrix requires only $\mathcal{O}(|\mathcal{V}|+|\mathcal{E}|)$ memory. (ii) *Pruned Search Space:* Gradients are computed only for the selected $\mathcal{S}_{leaf}$ nodes, limiting additional memory to $\mathcal{O}(B) \ll N$ and avoiding out-of-memory issues common in full-graph gradient attacks. **Time Complexity:** Standard iterative attacks (*e.g.*, PGD, PR-BCD) require K forward-backward propagations (*e.g.*, $K \in [30, 100]$). LPSA performs the attack in a single backward propagation ($K = 1$), achieving a speedup of nearly $K\times$ over iterative baselines.

TABLE II: Dataset statistics.

Dataset	#Nodes	#Edges	#Features	#Classes
Citeseer	3,327	9,104	3,703	6
Pubmed	19,717	88,648	500	3
arXiv	169,343	2,315,598	128	40
MAG	736,389	10,792,672	128	349

Overall, LPSA is highly efficient and scalable, making it practical for large-scale graphs.

V. EXPERIMENTS

In this section, we conduct comprehensive experiments to evaluate the effectiveness, transferability, and scalability of our proposed LPSA on widely used graph benchmarks. All attacks are conducted under a strict gray-box setting without querying victim models.

A. Experimental Setup

Datasets. We utilize four widely used benchmark datasets for node classification, covering graphs of different scales: Citeseer, Pubmed, arXiv, and MAG. For Citeseer and Pubmed, we adopt the publicly available splits following prior work [17]. For arXiv and MAG, we use the official splits provided by the Open Graph Benchmark.

Victim Models. To evaluate the transferability of our attacks, we select four representative GNNs as victim models: 1) **GCN** [17]: learns node representations by aggregating information from the local neighborhood via a first-order approximation of spectral convolutions. 2) **SGC** [18]: is a simplified linear variant of GCN that removes non-linear activation functions to reduce complexity while maintaining competitive performance. 3) **GAT** [19]: leverages a masked self-attention mechanism to assign learnable weights to different neighbors during aggregation. 4) **GraphSAGE** [20]: learns node embeddings by sampling and aggregating features from a fixed-size local neighborhood. It is worth noting that results for GAT on the MAG dataset are reported as N/A; this is due to Out-Of-Memory (OOM) errors during the training phase.

Baseline Attacks. We compare the proposed LPSA with four state-of-the-art attack methods: 1) **RA**: perturbs the graph by randomly flipping edges associated with the target node. 2) **DICE** [11]: randomly disconnects edges within the same class and connects edges between different classes. 3) **SGA** [14]: extracts the target node's subgraph and computes gradients exclusively within this local region to manipulate the graph structure. 4) **PR-BCD** [9]: leverages projected randomized block coordinate descent to optimize the discrete graph structure. We exclude exhaustive methods like Nettack [4] as their greedy search strategy is computationally infeasible for datasets larger than Cora.

Evaluation Protocol and Metrics. We follow the standard evaluation protocol for gray-box evasion attacks. For each dataset, we randomly sample 500 target nodes from the test set that are correctly classified by the victim models. The adversarial examples are generated solely based on the surrogate model. Subsequently, these generated perturbations are transferred to the target victim models to assess transferability.

TABLE III: Attack Success Rate (%) on large-scale datasets (*i.e.*, arXiv and MAG) under various attack budgets Δ. Bold indicates the best performance. (N/A: not applicable.)

Δ	Victim Model	arXiv					MAG				
		RA	DICE	SGA	PR-BCD	**Ours**	RA	DICE	SGA	PR-BCD	**Ours**
$\alpha = 0.10$	GCN	2.07±1.15	4.00±0.72	34.73±1.67	56.47±1.70	**63.13±3.23**	8.87±0.64	13.73±0.95	37.13±4.92	**80.78±0.83**	**81.20±3.10**
	SGC	2.60±1.11	4.33±1.63	53.53±2.91	54.73±3.10	**63.40±2.78**	8.93±1.50	12.27±1.29	33.8±3.98	**67.80±2.65**	**70.93±3.44**
	GAT	1.47±0.12	2.73±0.12	13.67±2.20	12.53±1.14	**15.47±0.95**	N/A	N/A	N/A	N/A	N/A
	GraphSAGE	1.67±0.99	2.87±0.64	13.00±0.53	15.27±1.17	**16.07±0.99**	4.87±0.55	7.60±0.69	7.77±1.40	**23.67±0.93**	**23.40±1.25**
$\alpha = 0.15$	GCN	3.67±0.23	4.53±1.14	50.87±1.42	75.40±1.40	**80.93±2.00**	11.73±2.41	18.07±2.81	47.47±1.80	**86.80±1.06**	**88.20±1.91**
	SGC	3.73±0.95	6.13±0.90	67.53±2.32	72.33±0.61	**79.67±2.16**	10.60±1.25	18.40±1.93	42.40±1.40	**80.47±1.55**	**79.93±3.64**
	GAT	2.27±0.58	3.33±1.10	20.53±1.79	21.53±1.01	**24.67±2.57**	N/A	N/A	N/A	N/A	N/A
	GraphSAGE	2.27±0.58	2.93±0.31	19.00±0.53	23.53±1.55	**25.47±2.08**	7.87±1.01	10.97±0.95	11.73±0.72	30.33±1.76	**34.70±1.76**
$\alpha = 0.20$	GCN	4.27±0.23	6.80±0.92	60.13±1.63	84.00±1.93	**87.67±2.48**	15.73±0.64	25.60±2.65	56.87±3.64	**90.87±1.55**	**90.80±0.72**
	SGC	4.07±0.90	6.53±1.45	77.00±1.93	83.20±2.08	**86.53±1.63**	13.67±1.50	21.20±0.87	52.27±1.50	86.40±0.80	**86.53±1.63**
	GAT	2.93±0.95	5.00±1.59	25.93±2.16	26.93±2.84	**33.20±4.68**	N/A	N/A	N/A	N/A	N/A
	GraphSAGE	3.00±0.60	4.87±0.70	24.93±1.68	30.20±0.92	**32.00±2.25**	9.73±1.19	14.03±0.91	13.80±1.32	33.77±1.79	**43.53±1.65**
$\alpha = 0.25$	GCN	5.00±0.92	8.00±1.83	70.67±1.27	90.80±0.72	**94.13±0.61**	17.33±1.62	30.87±1.75	64.67±2.34	**93.47±1.03**	92.27±0.61
	SGC	5.40±2.31	8.73±0.70	85.80±1.00	89.67±1.30	**93.13±0.64**	14.33±1.62	27.73±1.10	61.67±1.21	**91.07±1.42**	89.33±0.90
	GAT	4.73±0.50	5.80±0.72	32.53±0.83	34.40±1.71	**41.20±0.69**	N/A	N/A	N/A	N/A	N/A
	GraphSAGE	3.93±0.50	4.87±0.50	28.40±3.20	36.2±1.06	**39.73±1.42**	12.10±0.87	17.30±1.48	17.27±2.29	37.70±2.07	**49.00±0.70**

TABLE IV: Attack Success Rate (%) on small-scale datasets (*i.e.*, Citeseer and Pubmed) under various attack budgets Δ. Bold indicates the best performance. (N/A: not applicable.)

Δ	Model	Citeseer					Pubmed				
		RA	DICE	SGA	PR-BCD	**Ours**	RA	DICE	SGA	PR-BCD	**Ours**
$\alpha = 0.10$	GCN	2.30±1.99	2.30±3.98	11.74±1.78	**13.79±3.45**	**13.79±3.45**	2.05±1.31	1.78±0.93	16.91±0.66	**23.85±2.66**	**25.06±1.27**
	SGC	0.00±0.00	0.00±0.00	10.71±0.00	**17.86±0.00**	**17.86±0.00**	0.88±0.89	1.47±1.02	**19.47±0.00**	**19.76±0.51**	**19.76±0.51**
	GAT	0.00±0.00	2.15±1.86	3.23±3.23	**8.68±6.63**	**8.68±6.63**	1.52±1.39	4.26±5.23	16.65±3.86	**21.49±4.85**	**19.66±2.55**
	GraphSAGE	0.00±0.00	1.11±1.92	2.22±1.92	**6.75±0.13**	5.67±4.05	0.30±0.51	1.79±0.86	5.99±1.79	**7.49±1.95**	**7.19±2.27**
$\alpha = 0.15$	GCN	1.72±0.76	2.59±1.32	12.66±0.76	**18.46±2.05**	**18.46±2.05**	1.64±0.92	3.51±0.88	27.35±0.89	**31.66±1.49**	**31.23±2.23**
	SGC	2.60±1.30	2.60±2.25	13.04±1.23	**16.88±0.00**	**16.45±0.75**	1.66±1.30	3.31±0.95	**30.43±0.00**	**30.43±0.63**	**31.47±1.29**
	GAT	1.23±1.22	4.20±2.71	8.34±6.52	**14.20±7.49**	**13.36±7.12**	3.76±1.68	7.72±4.46	23.17±3.49	**27.95±3.97**	23.58±1.26
	GraphSAGE	0.85±1.48	0.85±1.48	3.77±0.10	**11.22±9.62**	**13.79±1.97**	1.66±0.73	3.11±0.02	12.24±0.94	**14.31±1.16**	**13.69±1.78**
$\alpha = 0.20$	GCN	0.86±0.86	2.03±1.00	12.95±2.74	**20.16±3.83**	**19.29±3.82**	1.60±0.54	6.55±2.31	35.54±0.13	**40.48±1.25**	**40.49±0.98**
	SGC	2.67±0.02	1.19±0.52	11.91±0.51	**16.62±1.35**	**15.73±0.48**	1.24±0.61	3.53±1.10	39.15±0.00	**40.92±0.31**	**40.74±1.40**
	GAT	2.24±2.35	4.24±2.89	9.06±5.76	**17.36±7.96**	**14.27±5.33**	3.06±2.06	8.23±3.87	28.50±5.31	**32.44±3.01**	**30.28±1.88**
	GraphSAGE	1.41±1.29	2.76±4.07	5.69±1.26	**15.68±1.87**	**14.23±1.54**	1.07±0.54	2.32±1.10	13.77±1.24	**17.35±1.57**	**16.09±2.21**
$\alpha = 0.25$	GCN	2.25±1.48	4.34±1.73	20.31±2.37	**30.78±4.72**	**30.02±4.38**	3.46±0.71	5.66±0.41	41.27±0.52	**50.55±0.35**	50.08±0.14
	SGC	2.74±1.47	3.33±1.48	21.77±1.56	**28.77±1.77**	**26.81±0.43**	1.09±0.27	4.36±0.27	47.66±0.00	**48.60±0.47**	**48.91±0.54**
	GAT	2.21±1.91	5.38±2.95	11.84±5.37	**21.74±8.65**	**17.71±6.64**	4.20±2.06	10.55±5.20	34.49±5.05	**40.66±4.70**	**37.72±2.53**
	GraphSAGE	1.85±0.30	2.40±1.25	9.64±0.72	**19.85±0.13**	18.55±0.39	2.50±0.28	3.75±0.02	16.84±1.52	**20.90±1.44**	18.87±1.55

To evaluate performance, we employ the Attack Success Rate (ASR) as the evaluation metric. ASR is defined as the percentage of target nodes that are successfully misclassified by the victim model after perturbation.

Parameter Settings. For the surrogate training phase of LPSA, we empirically set the Gaussian noise scale to $\sigma = 0.03$ and the softmax temperature to $T = 6$. The trade-off hyperparameter λ, which balances the classification loss $\mathcal{L}_{CE}$ and the smoothing regularization $\mathcal{L}_{smooth}$, is fixed to 1 across all datasets. In the attack phase, the confidence threshold τ for filtering candidate leaf nodes is set to 0.5. The search-space block size B is set to 4,000. For fair comparison, we set the block size of PR-BCD to the same value 4,000. For all other baseline attacks, we adopt the default parameter settings as recommended in the original papers [11], [14].

Implementation Details. To ensure a fair comparison, we standardize the surrogate model architecture across all methods. Specifically, we employ a 2-layer GCN with a hidden dimension of 256 and a dropout rate of 0.5. All baselines share the same pre-trained model weights θ, with the exception of SGA, which adopts an SGC as its surrogate consistent with its original implementation. For the optimization, the surrogate models are trained using the Adam optimizer with a learning rate of 0.01 for a maximum of 3,000 epochs, employing an early stopping mechanism with a patience of 300. Notably, when evaluating transferability across identical architectures (e.g., GCN $\rightarrow$ GCN), the victim models are trained using different random seeds from those of the surrogate models. For the attack phase, iterative baselines (i.e. PR-BCD) are run for 10 iterations to balance attack effectiveness and computational efficiency. In contrast, LPSA achieves competitive performance with a single iteration. Following the setting of PR-BCD [9], the perturbation budget is adaptive to the node degree, defined as $\Delta = \text{round}(\deg(v_t) \cdot \alpha)$. For a reliable evaluation, we report the mean and standard deviation over three random seeds.

B. Performance Evaluation

We report the attack results of all baseline methods against diverse victim models on diverse datasets. Tables III and IV summarize the attack success rates of all methods across large-scale and small-scale datasets, respectively, under various attack budgets.

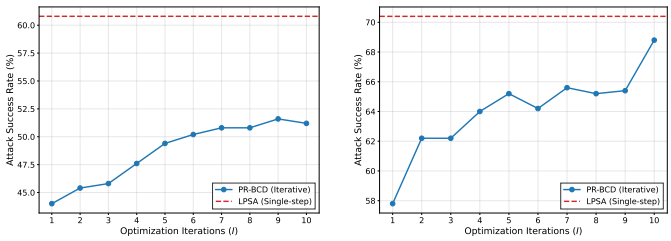
1) Comparison of Attack Effectiveness: **Performance Comparison on Large-scale Datasets.** The attack results of all baseline methods against diverse victim models on large-scale datasets are reported in Table III. As shown in Table III,

TABLE V: Comparison of runtime (seconds) on a GCN surrogate model.

Method	arXiv		MAG	
	Runtime (s)	Speedup	Runtime (s)	Speedup
PR-BCD ($I = 10$)	0.95	1.0×	5.49	1.0×
LPSA (Ours)	**0.20**	**4.75**×	**1.04**	**5.28**×

LPSA achieves the best or comparable performance among all the baseline methods for the large-scale datasets with lower computational cost. We vary the attack budget from $\alpha = 0.10$ to 0.25 to evaluate the attack effectiveness at different perturbation levels. Specifically, when attacking the GCN model with $\alpha = 0.25$, LPSA reaches an ASR of 94.13% on arXiv and 92.27% on MAG. On MAG dataset, LPSA improves the best competitor (i.e. PR-BCD) by 11.3% ASR against the GraphSAGE model ($\alpha = 0.25$). Even with a tighter budget ($\alpha = 0.10$), our method is competitive with the strongest competitor. For instance, against SGC, LPSA improves the ASR by 8.67% on arXiv and 3.13% on MAG compared to PR-BCD. LPSA remains effective when attacking more challenging victim models such as GAT and GraphSAGE, achieving higher ASR than PR-BCD. We note that the absolute ASR on GAT and GraphSAGE remains low, as their attention and sampling mechanisms inherently mitigate structural perturbations. Nevertheless, LPSA consistently achieves relative improvements over baselines by concentrating perturbations on structurally asymmetric leaf nodes.

Performance Comparison on Small-scale Datasets. We further evaluate the attack effectiveness on small-scale benchmarks, including Citeseer and Pubmed, and report the results in Table IV. Compared to traditional heuristic baselines (RA, DICE) and greedy methods (SGA), optimization-based attack (PR-BCD) achieve substantially higher ASR. On these smaller graphs, iterative optimization method can exhaustively explore the global search space, thereby establishing an empirical upper bound for attack effectiveness. Despite operating on a restricted leaf-prior search space and using only a single-step gradient update, LPSA achieves statistically comparable performance to PR-BCD across different victim models and attack budgets. In most settings, the ASR of LPSA closely matches that of PR-BCD, with differences falling within one standard deviation, indicating no statistically significant performance degradation. For example, on Citeseer against GCN ($\alpha = 0.20$), the results for PR-BCD ($20.16\% \pm 3.83$) and LPSA ($19.29\% \pm 3.82$) show substantial overlap in their performance variability. This pattern holds across a wide range of perturbation budgets ($\alpha = 0.10$ to 0.25) and victim architectures, including GCN, SGC, GAT, and GraphSAGE. Notably, while PR-BCD relies on iterative optimization to gradually refine perturbations, LPSA reaches a similar performance using a single-step gradient on a surrogate model. These results demonstrate that LPSA can effectively approximate the upper-bound performance of strong iterative attacks on small-scale datasets, while requiring substantially lower computational overhead. Such statistical ties with PR-BCD highlight the efficiency–effectiveness trade-off achieved by LPSA, and motivate its application to large-scale scenarios where iterative optimization becomes prohibitively expensive. Overall, on small-scale graphs where iterative optimization is feasible, LPSA achieves performance that is statistically indistinguishable from the strong iterative baseline (PR-BCD), while requiring only a single gradient step.



(a) arXiv (b) MAG

Fig. 2: Attack success rate vs number of iterations for PR-BCD compared to LPSA (single-step) on arXiv and MAG.

2) Efficiency and Scalability Analysis: We evaluate the efficiency and scalability of LPSA in comparison with the iterative PR-BCD method. While PR-BCD achieves high attack success rates through multiple iterative optimization steps, it requires significant computational cost that grows with graph size. In contrast, LPSA performs a single-step gradient update within the leaf-prior subspace, drastically reducing runtime and demonstrating strong scalability to large graphs. We report the computational efficiency by measuring the average runtime required to generate perturbations for a single target node. All runtime experiments are conducted on an NVIDIA RTX 4090 GPU. As summarized in Table V, LPSA reduces runtime by approximately $5\times$ on both arXiv and MAG, corresponding to an over 80% reduction in computational resources. This efficiency advantage becomes increasingly critical on large-scale graphs: while PR-BCD incurs substantial overhead due to repeated forward-backward passes, LPSA achieves competitive performance with a single backward propagation, demonstrating strong scalability.

To further evaluate the attack efficiency, we visualize the Attack Success Rate (ASR) against the number of optimization iterations (I) in Fig. 2. As illustrated in Fig. 2, the attack success rate of PR-BCD gradually improves as the number of iterations (I) increases, indicating that its effectiveness relies on repeated optimization steps. In contrast, LPSA achieves a competitive ASR with a single gradient update, still matching or exceeding PR-BCD at I=10 on both arXiv and MAG datasets. This comparison highlights that LPSA effectively approximates the performance of iterative attacks at a fraction of the computational cost, which is particularly important for large-scale graphs.

Overall, LPSA offers a computationally efficient approximation of upper-bound iterative attacks, achieving an improved trade-off between effectiveness and computational cost. Its single-step design ensures scalability to large-scale datasets, making it practical for real-world adversarial scenarios.

C. Ablation Study

To investigate the contribution of each component in LPSA, we conduct ablation studies on the arXiv dataset under

TABLE VI: Ablation study on arXiv dataset ($\alpha = 0.1$).

Method	GCN	SGC	GAT	GraphSAGE
w/o Leaf-Prior	43.80±1.06	45.87±3.37	10.60±1.39	12.07±1.85
w/o Fine-tuning	**63.40±2.42**	**62.20±2.25**	**15.20±0.72**	**15.53±1.14**
LPSA	**63.13±3.23**	**63.40±2.78**	**15.47±0.95**	**16.07±0.99**

$\alpha = 0.10$. We consider two variants: (i) *w/o Leaf-Prior*, which maintains the same candidate set size but samples perturbation nodes uniformly at random instead of restricting them to leaf nodes; and (ii) *w/o Fine-tuning*, which generates adversarial perturbations using a vanilla surrogate model without manifold-smoothed fine-tuning. The attack success rates (ASR) are reported in Table VI.

Effectiveness of the Leaf-Prior Subspace. As shown in Table VI, removing the leaf-prior constraint results in a substantial drop in attack success rate across all victim models. For example, on GCN, the ASR decreases from 63.13% to 43.80%, indicating that randomly sampled nodes are significantly less effective than leaf nodes. This supports our insight that structural asymmetry renders leaf nodes particularly effective for attacks. By targeting this privileged subset, LPSA bypasses inefficient global search while maintaining strong effectiveness.

Impact of Score Matching Fine-tuning. We observe that the vanilla surrogate achieves slightly higher ASR when attacking GCN with the same architecture, which can be attributed to its gradients being highly specialized to GCN's inductive bias. However, such architecture-specific gradients exhibit limited transferability. By incorporating manifold-smoothed fine-tuning, LPSA sacrifices a small amount of specificity in the homogeneous setting (-0.27% on GCN) for improved gradient transferability across heterogeneous victim models (e.g., $+1.20\%$ on SGC and $+0.54\%$ on GraphSAGE). Crucially, in a single-step attack setting where the perturbation relies entirely on the gradient of the surrogate model, this enhancement in gradient quality ensures reliable adversarial edge selection across diverse architectures.

VI. Conclusion

This paper proposes LPSA, a highly efficient leaf-prior structural attack framework for large-scale graph data. By identifying the structural asymmetry and fragility of leaf nodes, we design a leaf-prior perturbation space that substantially reduces the complexity of discovering effective adversarial edges. To improve gradient transferability in the single-step setting, we further introduce a manifold-smoothed surrogate fine-tuning strategy, which aligns adversarial gradients with the intrinsic data distribution. Extensive experiments demonstrate the effectiveness of LPSA across diverse datasets and victim architectures. In particular, LPSA achieves competitive or superior attack success rates compared to state-of-the-art iterative baselines, while reducing runtime by approximately 80%, highlighting its strong efficiency–effectiveness trade-off for large-scale graph attacks. Extending the theoretical analysis beyond mean aggregation and to denser graph structures remains future work.

References

[1] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 34, no. 5, pp. 2033–2047, 2020.

[2] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in Genetics*, vol. 12, pp. 690049, 2021.

[3] Shiwen He, Shaowen Xiong, Yeyu Ou, Jian Zhang, Jiaheng Wang, Yongming Huang, and Yaoxue Zhang. An overview on the application of graph neural networks in wireless networks. *IEEE Open Journal of the Communications Society*, vol. 2, pp. 2547–2565, 2021.

[4] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2847–2856, 2018.

[5] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 3389–3396, 2020.

[6] Zhaohan Xi, Ren Pang, et al. Graph backdoor. In *Proceedings of the USENIX Security Symposium*, pages 1523–1540, 2021.

[7] Zulfikar Alom, Tran Gia Bao Ngo, Murat Kantarcioglu, and Cuneyt Gurcan Akcora. Gottack: Universal adversarial attacks on graph neural networks via graph orbits learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–14, 2025.

[8] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 2, pp. 19–34, 2021.

[9] Simon Geisler, Tobias Schmidt, Hüseyin Sirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 7637–7649, 2021.

[10] Geng Zhu, Ming Chen, Chun Yuan, and Yan Huang. Simple and efficient partial graph adversarial attack: A new perspective. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 36, no. 8, pp. 4245–4259, 2024.

[11] Marcin Waniek, Tomasz P. Michalak, Talal Rahwan, and Michael Wooldridge. Hiding individuals and communities in a social network. *Nature Human Behaviour*, vol. 2, pp. 139–147, 2018.

[12] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3961–3967, 2019.

[13] Xu Liu, Jing-Jing Huang, and Wei Zhao. Gcia: A black-box graph injection attack method via graph contrastive learning. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6570–6574, 2024.

[14] Jia Li, Tianchi Xie, Ling Chen, Fei Xie, Xiangnan He, and Zibin Zheng. Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 35, no. 1, pp. 82–95, 2023.

[15] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[16] Yao Zhu, Yuefeng Chen, Xiaodan Li, Kejiang Chen, Yuan He, Xiang Tian, Bolun Zheng, Yaowu Chen, and Qingming Huang. Toward understanding and boosting adversarial transferability from a distribution perspective. *IEEE Transactions on Image Processing (TIP)*, vol. 31, pp. 6487–6501, 2022.

[17] TN Kipf. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–14, 2017.

[18] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 6861–6871, 2019.

[19] Petar Veličković, Guillem Cucurull, Arantxa Casanova, et al. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.